Homework 2

b03901027 徐彥旻

Handwriting

1. Encryption Algorithms

對稱式與非對稱式加密不同的地方：1) 由於非對稱式加密系統加、解密時使用的金鑰不同，可以進行數位簽章，而對稱式金鑰則因為加解密所使用金鑰相同，故無法用簽章來實作不可否認性。此外，2) 對稱式金鑰在分配時需要 $O(n^2)$ 數量的金鑰，但是由於非對稱式金鑰的公鑰大家都可以用，故只需要 $O(n)$ 數量的金鑰即可，其中 n 為 device 的數量或是人數。

對稱式的例子：DES, 3DES, AES

非對稱式的例子：RSA, ECC

下圖為課本 *Understanding Crytography* 中各類型加密的金鑰長度與安全程度的表格。由下圖中可以看出，若要達到相似的安全程度，非對稱式密碼的金鑰都會比對稱式密碼來得長。一般來說，對稱式密碼會比較快，因為其運算是由「置換、查表、互斥或」構成，而非對稱式密碼的運算則是由次方與乘法的模運算構成。

Table 6.1 Bit lengths of public-key algorithms for different security levels

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

2. Three-way Diffie-Hellman

Alice, Bob, and Charlie 分別生成他們自己的秘密 k_A, k_B, k_C, 計算 $A = g^{k_A} \bmod p$, $B = g^{k_B} \bmod p$, $C = g^{k_C} \bmod p$。接著，Alice 將 A 傳給 Bob, Bob 將 B 傳給 Charlie , Charlie 將 C 傳給 Alice。經過這一輪傳送後，Alice, Bob, and Charlie 分別計算 $AC = C^{k_A} \bmod p$, $BA = A^{k_B} \bmod p$, $CB = B^{k_C} \bmod p$, 然後 Alice 將 AC 傳給 Bob, Bob 將 BA 傳給 Charlie , Charlie 將 CB 傳給 Alice。

最後，Alice, Bob, and Charlie 分別計算 $ACB = CB^{k_A} \bmod p$，$BAC = AC^{k_B} \bmod p$，$CBA = BA^{k_C} \bmod p$。這三者會相等，即為 Alice, Bob, and Charlie's share secret，他們可以用這個共享的秘密來生成金鑰。

3. ElGamal threshold decryption

setup:

a trusted third party : TTP

large prime : p

generator : g

secret key : $sk_B = b$

public key : $pk_B = g^b \ (mod\ p)$

compute a random degree $t-1$ polynomial:

$$f(x) = b + \sum_{j=1}^{t-1} a_i x^i \ (mod\ p)$$

encryption:

plaintext : m

random value : k

ciphertext1 : $c_1 = g^k \ (mod\ p)$

ciphertext2 : $c_2 = m(pk_B)^k \ (mod\ p)$

decryption:

TTP compute n shares of b: $s_i = g^{xf(x_i)} \ (mod\ p)$ for each user i

compute Λ_i values,

where $\Lambda_i = L_i(0) = \prod_{j \in I,\, i \neq j} \frac{x_j}{x_j - x_i} \bmod p$,

where I is any size t subset of $\{1,\ n\}$

compute g^{ka},

$$g^{ka} = \prod_{i \in I} s_i^{\Lambda_i} = g^{k \sum_{i \in I} f(x_i)\Lambda_i} \equiv g^{kf(0)} \equiv g^{ka} \ (mod\ p)$$

plaintext : $m = c_2(g^{ka})^{-1} = (mg^{ka})g^{-ka} = m \ (mod\ p)$

在這樣的設定中，需要 t 位使用者合作才能計算出 Λ_i 以及 g^{ka}，是個 (t, n) threshold scheme 。

Capture The Flag

4. ECB Encryption Mode

Flag: BALSN{W0w_y0u_4r3_r3411y_4_cu7_4nd_p4st3_m4st3r}

目標是做出 'login=' || name || '&pwd=' || pwd || '&role=admin'

而現有能操作的是 'login=' || x || '&role=user' || '&pwd=' || y

step 1: 讓 len(x) = 20，得到 'login=' || x_1 以及 x_2 || '&role=' 加密後的結果

其中 x_1 = x[:10]，x_2 = x[10:]

step 2: 讓 len(x) = 11, y = 'admin'，得到 'admin' 加密後的結果（有 padding ）

step 3: 讓 len(x) = 16, len(y) $\geq$ 11，得到 '&pwd=' || y_1 加密後的結果

其中 y_1 = y[:11]

step 4:

將加密的結果組合成 'login=' || x_1 || '&pwd=' || y_1 || x_2 || '&role=' || 'admin' 加密後的結果，即為 token。而 name 即為 x_1、pwd 即為 y_1 || x_2。

5. Beginner's RSA

(1) Flag: BALSN{V3RYW311}

利用指令 openssl rsa -text -inform PEM -in public.pem -pubin 查看 n 和 e，將 n 轉換為十進制的數字之後，使用線上服務 factordb.com 對其做質因數分解，得到 $p, q \ \ s.t. \ p * q = n$，可得尤拉函數的值，再利用 extended Euclidean algorithm 計算出對應的密鑰 d，即可解密 flag.enc 。

(2) Flag: BALSN{Forty Years of Attacks on the RSA Cryptosystem}

$d_1 * e_1 \equiv 1 \ (mod\ \phi(n))$ 可以寫成 $d_1 * e_1 = 1 + k_1 * \phi(n)$，由於此題的 $e_1 = 7$，又 d_1 應小於 $\phi(n)$，可知 $k_1 \in \{1, 2, 3, 4, 5, 6\}$。由與 k_1 只有六種可能，可以窮舉所有的 $\phi(n) = (d_1 * e_1 - 1) / k_1$。計算 $d_A = e^{-1} \ (mod\ \phi(n))$ 得到 Alice 的密鑰，將密文解密，結果若為 flag 的格式則代表解出答案。

(3) Flag: BALSN{Keep calm and count prime numbers}

因為使用的大數 n 相同，且加密的明文相同，適用於 Common modulus attack [1] 的情境，利用 extended euclidean algorithm 找出 s_1 與 s_2 滿足 $e_A * s_1 + e_B * s_2 = 1$，其中 e_A, e_B 分別為 Alice 與 Bob 的公鑰。（實作上因為會有其中一個 s 為負數，假設是 s_1，需要先用 extended euclidean algorithm 找到 e_A 的 inverse，再做次方的模運算）

如此一來，就可以利用 $C_A{}^{s_1} * C_B{}^{s_2} \equiv (C_A{}^{-1})^{s_1} * C_B{}^{s_2} \equiv M \ (mod\ n)$ 這個數學關係來求出明文。

Reference:
[1]https://crypto.stackexchange.com/questions/16283/how-to-use-common-modulus-attack

6. Digital Certificate

Flag: BALSN{b451c_s3!f_51gn3d_c3rt1fic4t3}

[2] 提供了詳細的流程，只要依照順序執行即可生成 fake.crt，再用 python 將其讀入，以 base64 編碼後，傳至 140.112.31.109 10005，即可得到 Flag。以下簡短說明重要的步驟：

步驟一：產生密鑰

openssl genrsa -des3 -out server.key 1024

在此步驟中，會輸入自訂的 PEM　　pass phrase

步驟二：產生「簽章請求」

openssl req -new -key server.key -out server.csr

接著輸入題目所要的資訊

步驟三：移去 passphrase

cp server.key server.key.org

openssl rsa -in server.key.org -out server.key

步驟四：生成自簽簽證

openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt

做到這步所生成的 server.crt 即為本題要的簽證。

Reference:
[2]http://www.akadia.com/services/ssh_test_certificate.html

7. I need your help

Flag: BALSN{Now_you_know_the_secret}

[3] 解釋了 key.pem 的裡頭有 $n, e, d, p, q, dp, dq, qinv$ 等資訊。[4] 解釋這個檔案放的格式，並有長度符合本題的例子（2048 bit），參考其分隔方式，便可以解讀沒有被遮住的資訊，得到 $e, p, q, dp, dq, qinv$。

解法一：

以 extended euclidean algorithm 算出密鑰 （$d = e^{-1} \ mod \ \phi(p * q)$），將密文做 d 次方的模運算，即可得到明文。

解法二：

已知 p, q, dp, dq 可列出如下的式子：

$$d \equiv dp \ mod \ (p - 1)$$

$$d \equiv dq \ mod \ (q - 1)$$

其中 d 為未知。利用中國剩餘定理，可解得 d。將密文做 d 次方的模運算，即可得到明文。

Reference:
[3]https://crypto.stackexchange.com/questions/21102/what-is-the-ssl-private-key-file-format

[4]https://etherhack.co.uk/asymmetric/docs/rsa_key_breakdown.html

8. I will look for you, and I will find you

(1) BALSN{Don't underestimate the power of the Dark Web}

使用 tor browser 連到 ab6qqnuetobsjslu.onion:31337 即可。

(2) BALSN{128.199.198.162}

使用 tor browser 連到 https://ab6qqnuetobsjslu.onion:10443， 查看憑證，在 Certificate > Issuer 中發現 E = admin@servers.tw， 利用任一線上的域名-IP查詢服務搜尋 servers.tw 之 IP， 可得 128.199.198.162， 之後再連過去一次確認內容一樣即可。

(3) BALSN{140.112.31.96}

使用指令 torify ./ssh -o FingerprintHash=md5 -o HostKeyAlgorithms=ssh-rsa ztczadd4tipwhwyl.onion 可得該服務在 rsa 之下的 md5 fingerprint， 這也是 Shodan 會存的資料種類，故可以拿此 Fingerprint 到 Shodan 搜尋，得到 IP。

(4) BALSN{104.198.2.240}

```
torsocks ./ftp  7zysy3slgt7qxhek.onion
ftp> rstatus
```

rstatus 會吐出 remote server 相關的資訊，這些資訊之中及包含了這個 server 的 IP。

(5) Not solved yet. GG

Acknowledgements

江緯璿同學